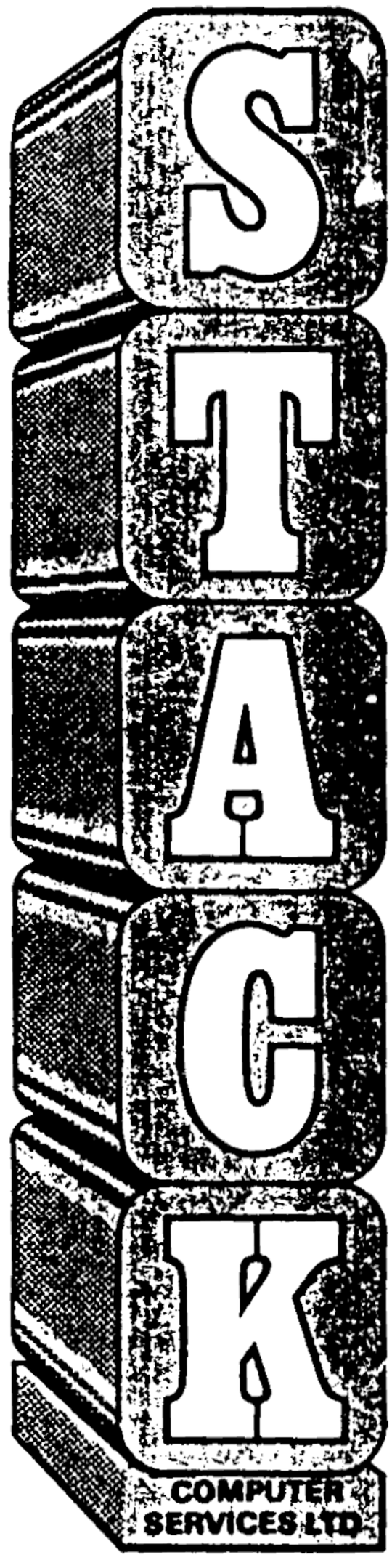
# VIC KIT 5
# USERS GUIDE



## INTRODUCTION.

VICKIT 5 is a 4K EPROM containing firmware which gives the VIC-20 a full 6502 assembler. VICKIT 5 will only work in conjunction with VICKIT 4.

VICKIT 5 will also run via a ROM SWITCHBOARD with the VICKIT I (programmers aid), VICKIT 3 (Hi-res) or (combination of 1 and 3).

The recommended board to carry VICKIT 5 is the STACK ROM SWITCHBOARD. It is possible other manufacture unit will accept VICKIT 4 and 5. The user must check that the unit has space for 2 off 2532 compatible 4k sockets, addressed $ A000 and $ B000 $ A000 for VICKIT 4. $ B000 for VICKIT 5.

## SOFTWARE.

The assembler may be used on any Vic regardless of any memory size with cassette or diskettes. The amount of memory you have determines the length of your source code with about 400 lines possible in a 8k machine and 1600 possible in a 32k. A printer is desireable but not mandatory.

The purpose of an assembler is to convert source code (mnenomic code) into the native language (machine code) of a particular type of computer. Using an assembler alleviates the programmer of tedious address calculations and looking up operation codes etc. Source code can be easily modified at any time and the program re-assembled without having to re- calculate everything as when coding is pure machine code.

The source code for the assembler is keyed in. line by line, in the same manner as a BASIC program. This means that SAVEing, LOADing and modification of your source program may be done as if the source were BASIC program.

The object code generated by this assembler is temporairily held in the top portion of memory (last address equal to the contents of $37/38 minus one)and occupies a multiple of 256 bytes as defined when you initiate execution (see chapter on Execution) and may be moved during an assembly. At the end of each assembly you will be asked whether not you wish to transfer the object code to its execution address.

You may request either a symbol table listing or a full cross reference were the labels used are listed in alphabetical order with there associated value and each line number where the label was used. This is extremely useful when modifying a source program, particularly if the program is rather long.

FITTING VICKIT 5.

a). To stack Rom Switchboard.

Remove the covers of the Rom Switchboard by pressing out the plastic rivots. This will reveal four 24 pin sockets. VICKIT 4 is designed to fit into either IC6 or IC8 If the RAM area of the Vic is fully populated use socket IC6. If not then IC8 should be used.

Locate the notched end of the VICKIT 4. Orientate the VICKIT 4 over the required socket with the notch closest to the gold fingers. Now gently push the chip home into the socket, making sure every pin sits correctly.

The board can now be plugged into the VIC. Assuming Vickit 4 is present. If the chip does not repond try the change over poke for the Rom Switchboard.

When you are happy with the installation replace the covers on the Rom Switchboard.

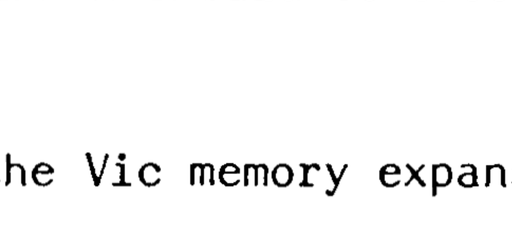
b). To a Supercharger.

Again remove the plastic rivots and covers from supercharger. Located inside are two 24 pin sockets VICKIT 4 should be fitted into the socket closest to the gold fingers i.e. B A 1. The notch on the EPROM should be closest to the two intergrated circuits IC1 and IC2.

Replace covers and plug into the Vic memory expansion. VICKIT 5 will function assuming VICKIT 4 is present.

c). To other type boards.

First check with the manufacturer that the socket is 4k, 2532 compatible and addressed 45056 to 49152 or Hexadecimal B000 to BFFF. Next locate pin 1 of the socket.
To locate pin 1 of the VICKIT place the VICKIT on table resting on its 24 pins, orientate the VICKIT so that the notch is on left hand side(TOP VIEW). Pin 1 is the pin nearest the viewer on the left most side. (BELOW THE NOTCH).




NOTE POWER SHOULD ALWAYS BE OFF WHEN PLUGGING ANY DEVICE IN OR OUT.

PSEUDO OPERATIONS.

All pseudo op codes begin with a period and may contain up to four characters but only the first character is needed and only the first two are output.
.B may be followed by any number of operands or expressions separated by commas and results in one byte of object code per operand being assembled. If a label or label expression is used, the value assembled will be the low order eight bits after evaluation.
.W is similar to .B except that each expression results in a double byte result with the low order byte stored first.
.D followed by an expression is used to define a block of memory of any length. Note that the memory defined is not initialized to any particular value.
.O followed by an expression (any symbols used must have been previously defined) may be used to set the location counter (memory address) to whatever value you desire. You may have several .O statements in program but the lowest value must occur first and the overall length of the program, including any gaps, must fit within the object code buffer defined at execution time. Also, the last .O instruction must be the highest value used for a program counter change.
.L is used to list the following lines on the output device. When the program is loaded the list function is on. Assembly errors are always listed on the output device.
.X is used to cancel the list function on the output device for the following lines.
.S is used if a symbol table (symbol and their addresses) is to be output at the end of the program.
.C is used if a sorted cross reference is to be putput at the end of the program. The cross-reference consists of the line number where each symbol isused preceded by the symbol name, in alphabetical order, and symbol value. If .C is specified as well as .S only the cross-reference will be output. Note that for very large source programs it may not be possible to enter the .C function since 4 bytes are added to the symbol table per reference.

SYNTAX.

All source instructions are keyed in after the line number. There are five zones, separated by at least one space, as follows:-

1. Line Number. The line number may be any number from 1 to 63999. Note that the numbers keyed in (or obtained via←N) are for your use in editing source - the assembler listing will contain its own sequential line numbers beginning at 1.

2. Label. The label, if needed for the line, is keyed in right after the line number. It may contain from 1 to 9 alphanumeric characters but must not contain the plus or minus sign nor start with a period. A label may not match a mnemonic instruction. For example, LDA will not be considered as a label.
NOTE THAT ZERO PAGE LABELS MUST BE DEFINED BEFORE BEING USED.

3. Mnemonic or pseudo op code. The mnemonic instruction is selected from the following list and follows the MOS Technology standard:-

ADC AND ASL BCC BCS BEQ BIT BMI BNE
BPL BRK BVC BVS CLC CLD CLI CLV CMP
CPX CPY DEC DEX DEY EOR INC INX INY
JMP JSR LDA LDX LDY LSR NOP ORA PHA
PHP PLA PLP ROL ROR RTI RTS SBC SEC
SED SEI STA STX STY TAX TAY TSX TXA
TXS TYA.
See also the chapter on pseudo

4. Operand. The operand also follows the MOS Technology standard. For immediate instructions or elements of a .B or .W instruction (see chapter on Pseudo Operations),the following conventions are used:-

a. A decimal number may be specified without a prefix.
b. The $ prefix specifies a hexadecimal number.
c. The greater than and less than symbols specify the high order or low order 8 bits of an expression.
d. The % sign followed by a string of up to eight 0's and 1's indicates a binary value.
e. The' (apostrpohe) prefix may be used to define a one character ASCII code. By using multiple occurances of this prefix in a .B statement you can enter messages to be displayed, for example.
f. Labels previously defined may be used in expressions including combinations such as label ± label ± decimal value.

PRINTED OUTPUT.

When you are satisfied with the results of an assembly, you may specify output to the printer by simply keying in OPEN 4.4:CMD 4 before you begin execution. This changes the output device which is normally the screen to be directed to a printer as device number 4 on the IEEE-488 bus.

This assembler outputs lines of up to 200 characters followed by a carriage return at the end of each line. In order for you to adapt non-standard printers, you can place a jump instruction in addresses $3F9-3FB to enter your own coding. Before each character is output, a call to the routine at the user vector $3F9-3FB is made with the character in the accumulator. To convert one character for example from a carriage return to a line feed, simply change the value in the accumulator and do a return (RTS) instruction. If you wish to add characters to the output you can accomplish this by doing a JSR $FFD2 with each character to be added in the accumulator and then a final return to the assembler.

Note that this vector can also be used to paginate the output, for example.

Note that if you are using the output vector(see chapter on Printer Output) you must use the ← B command rather than the standard ← E command to begin execution.

The first thing that will happen after you give the ← E or ← B command is that a message will appear asking how many B(blocks)(multiples of 256) you require for your object code to which you may answer from 0 to 9. This will reserve a block of memory with the end of the object code buffer equal to the top of memory($37/38 minus 1).

During the first pass of the assembler one position in the top line of the screen will change frequently to show that the program is running. The second pass results in the listing being output to the screeen or printer. Note that the assembly process may be suspended indefinately by depressing any key one time and then resumed by pressing another key.

1. Address of object code.
2. Object code in hexadecimal.
3. Sequential line number beginning at 1.
4. Label.
5. Mnemonic or pseudo operation code.
6. Operand.
7. Comments. Note that comments are supressed if the operand is more than 27 characters long.

Continued.....

During an assembly, any of the following error messages may be displayed if there are any errors in your source program;

1. Program counter expression error.
2. Duplicate label.
3. Format error.
4. Invalid mnemonic.
5. Invalid operand.
6. Object code averflow.
7. Branch out of range.
8. Too many symbols.
9. Invalid symbol.

You will normally have to correct your source code and re-assemble upon occurance of any of these errors.
At the end of the assembly another message will occur at the bottom of the display screen asking you if you wish to move the object code to its execution address or not. Once you have the object code at its execution address, you may initiate execution via the G function of the built-in monitor or via SYS call from BASIC. You may save your object code on tape using the ARROW or built in monitor.

```
#B? 1

                        1               .C
1200                    2               .OR $1200
1834                    3  FIELD       =   $1834
008A                    4  ZPAGE       =   $8A
7800                    5  PART2       =   $7800
0014                    6  OFFSET      =   20
                        7  ;*****************
                        8  ;ADDRESSING MODES
                        9  ;*****************
1200    69F4           10               ADC $F4          ;IMMEDIATE
1202    6D3418         11               ADC FIELD        ;ABSOLUTE
1205    658A           12               ADC ZPAGE        ;ZERO PAGE
1207    0A             13               ASL A            ;ACCUMULATOR
1208    18             14               CLC              ;IMPLIED
1209    618A           15               ADC (ZPAGE,X)    ;PRE-INDEXED INDIRECT
120B    718A           16               ADC (ZPAGE),Y    ;POST INDEXED INDIRECT
120D    758A           17               ADC ZPAGE,X      ;ZERO PAGE INDIRECT
120F    7D3418         18               ADC FIELD,X      ;ABSOLUTE X INDEXED
1212    793418         19               ADC FIELD,Y      ;ABSOLUTE, Y INDEXED
1215    9003           20               BCC *+5          ;RELATIVE
1217    6C3418         21               JMP (FIELD)      ;INDIRECT
121A    B68A           22               LDX ZPAGE,Y      ;ZERO PAGE INDEXED BY Y
                       23  ;
                       24  ;*****************
                       25  ;OPERAND SAMPLES
                       26  ;*****************
121C    A90F           27                   LDA 15       ;DECIMAL
121E    A915           28                   LDA $15      ;HEXADECIMAL
1220    A918           29                   LDA >FIELD   ;HIGH ORDER 8 BITS
1222    A934           30                   LDA <FIELD   ;LOW ORDER 8 BITS
1224    09DA           31                   ORA %11011010    ;BINARY
1226    9005           32                   BCC LOOP+3       ;LABEL WITH DISPLACEMENT
1228    B00E           33                   BCS LOOP+OFFSET-6    ;COMPLEX LABEL
122A    4C3418         34  LOOP             JMP FIELD
122D    0234FE         35                   .BY 2,<FIELD,$FE     ;MULTIPLE BYTE
1230    544558         36                   .BY 'T,'E,'X,'T      ;ASCII VIA .BY
1234    3418           37                   .WO FIELD            ;WORD
1236    4578           38                   .WO $7845,10         ;MULTIPLE BYTE EXPRESSION

FIELD       1834    11    18    19    21    29
                    30    34    35    37
LOOP        122A    32    33
OFFSET      0014    33
PART2       7800
ZPAGE       008A    12    15    16    17    22

    0 ERRORS, 14BF FREE

MORE Y/N? N

READY.
```

## DEMONSTRATION OF ASSEMBLER.

```
100 ; VICKIT 5
110 ; DEMONSTRATION OF ASSEMBLER OPERATING MODES
120 ;                    COMMENT
130 .O 16000           ;ORIGIN
140 ADR=456            ;LABEL
150 .C                 ;CROSSREF
160 TXA                ;ORG IMPLIED
170 LDA 123            ;0+1 IMMEDIATE DECIMAL
180 LDA %111           ;0+3 IMM BINARY
190 LDA $FF            ;0+5 IMM HEX
200 LDA ADR            :0+7 ABSOLUTE
210 LDA ADR,X          ;0+10 ABS+X
220 LDA ADR,Y          ;0+13 ABS+Y
230 ZER=123
240 LDA ZER            ;0+16 ZERO PAGE
250 LDA (ZER,X)        ;0+18 IX INDEXED INDIRECT
260 LDA (ZER),Y        ;0+20 IY INDIRECT INDEXED
270 LDA ZER,X          ;0+22 ZPG+X
280 LDY ZER,X          ;0+24 ZPG+X
290 LDX ZER,Y          ;0+26 ZPG+Y (ZPG+Y IS FOR LDX ONLY)
300 JMP ADR            ;0+28 ABSOLUTE
310 JMP (ADR)          ;0+31 INDIRECT (IND IS FOR JMP ONLY)
320 ROR ZER            ;0+34 ZERO PAGE
330 ROR A              ;0+36 ACCUMULATOR
340 ROR ADR            ;0+37 ABSOLUTE
350 .B 123             ;0+40 DATA BYTE OR BYTES
360 .W 456             ;0+41 DATA WORD OR WORDS
370 RTS                ;0+43 REMEMVER THE END
380 ;PRINTER MUST BE OFF AT POWER-UP
390 ;THEN OPEN 4,4:CMD4:SYS45456
400 ; \ 100,10         NUMBER
410 ; Q                QUIT NUMBERING
420 ; R 100,10         RENUMBER
430 ; B OR L OR SYS45056 ASSEMBLE
```

```
 B? 8

                    1 ; VICKIT 5
                    2 ; DEMONSTRATION OF ASSEMBLER OPERATING MODES
                    3 ;                   COMMENT
3E80                4                .O   16000          ;ORIGIN
01C8                5 ADR            =    456            ;LABEL
                    6                .C                  ;CROSSREF
3E80 8A             7                TXA                 ;ORG IMPLIED
3E81 A97B           8                LDA 123             ;0+1 IMMEDIATE DECIMAL
3E83 A907           9                LDA %111            ;0+3 IMM BINARY
3E85 A9FF          10                LDA $FF             ;0+5 IMM HEX
3E87 ADC801        11                LDA ADR             ;0+7 ABSOLUTE
3E8A BDC801        12                LDA ADR,X           ;0+10 ABS+X
3E8D B9C801        13                LDA ADR,Y           ;0+13 ABS+Y
007B               14 ZER            =    123
3E90 A57B          15                LDA ZER             ;0=16 ZERO PAGE
3E92 A17B          16                LDA (ZER,X)         ;0+18 IX INDEXED DIRECT
3E94 B17B          17                LDA (ZER),Y         ;0+20 IY INDIRECT INDEXED
3E96 B57B          18                LDA ZER,X           ;0+22 ZPG+X
3E98 B47B          19                LDY ZER,X           ;0+24 ZPG+x
3E9A B67B          20                LDX ZER,Y           ;0+26 ZPG+Y (ZPG+Y IS FOR LD
X ONLY)
3E9C 4CC801        21                JMP ADR             ;0+28 ABSOLUTE
3E9F 6CC801        22                JMP (ADR)           ;0+31 INDIRECT (IND IS FOR J
MP ONLY)
3EA2 667B          23                ROR ZER             ;0+34 ZERO PAGE
3EA4 6A            24                ROR A               ;0+36 ACCUMULATOR
3EA5 6EC801        25                ROR ADR             ;0+37 ABSOLUTE
3EA8 7B            26                .B   123            ;0+40 DATA BYTE OR BYTES
3EA9 C801          27                .W   456            ;0+41 DATA WORD OR WORDS
3EAB 60            28                RTS                 ;0+43 REMEMBER THE END
                   29 ;PRINTER MUST BE OFF AT POWER-UP
                   30 ;THEN OPEN 4,4:CMD4:SYS45056
                   31 ; N 100,10     NUMBER
                   32 ; Q            QUIT NUMBERING
                   33 ; R 100,10     RENUMBER
                   34 : B OR E OR SYS45056 ASSEMBLE

ADR    01C8 11   12   13   21   22
            25
ZER    007B 15   16   17   18   19
            20   23

   0 ERRORS, 416B FREE

MOVE Y/N? Y

READY.
```

DISCLAIMER.

Whilst every effort has been made to provide a flexible , reliable and above all low-cost product STACK COMPUTER SERVICES LTD. wish to point out that no claim is made for complete compatibility with any other equipment or program. The information given is believed to be accurate but no liability can be accepted for the consqyences of any error. Ours is a policy of continued development and we therefore reserve the right to alter the design of sppecifications without prior notice.

REQEST FOR INFORMATION.

In order to provide the user with as much support as is practical, we would appreciate if any useful comments or hints could be forwarded in to writing to:-

PRODUCT DEVELOPMENT
STACK COMPUTER SERVICES LTD
290/298 DERBY ROAD
BOOTLE,
LIVERPOOL L20 8LN.

(c) STACK 82.

Copyright
**Stack Computer Services Limited**
290-298 Derby Road, Bootle, Liverpool L20 8LN,
England.